Wintersemester 2017/18
Prof. Dr. J. Vygen
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik
3. Übung

1. Sei G ein zusammenhängender ungerichteter Graph mit Kantengewichten $c : E(G) \to \mathbb{R}$. Dabei seien alle Kantengewichte verschieden, also $c(e) \neq c(e')$ für $e \neq e'$.

   (a) Zeigen Sie, dass es dann genau einen kostenminimalen aufspannenden Baum T in G gibt.

   (b) Ein zweitbester aufspannender Baum sei ein aufspannender Baum, der von T verschieden ist und unter allen von T verschiedenen aufspannenden Bäumen kleinste Kosten hat. Zeigen Sie, dass es mehrere zweitbeste aufspannende Bäume geben kann.

   (c) Geben Sie einen möglichst effizienten Algorithmus zur Berechnung eines zweitbesten aufspannenden Baums an. Zeigen Sie die Korrektheit Ihres Verfahrens. (2+2+2 Punkte)

2. Betrachten Sie folgendes Problem: Gegeben seien ein ungerichteter zusammenhängender Graph G mit Kantengewichten $c : E(G) \to \mathbb{R}$, ein Knoten $v_0 \in V(G)$ und eine Zahl $k \in \mathbb{N}$ mit $k \leq |\delta_G(v_0)|$. Gesucht ist ein aufspannender Baum von T in G, so dass v_0 in T mindestens Grad k hat, der unter allen aufspannenden Bäumen in G, in denen v_0 mindestens Grad k hat, minimales Gewicht hat. Geben Sie einen Algorithmus mit polynomieller Laufzeit an, der dieses Problem löst. (5 Punkte)

3. (a) Zeigen Sie, dass es Folgen von Heap-Operationen gibt, so dass in einem Fibonacci-Heap die maximale Pfadlänge in einer Arboreszenz $\Theta(n)$ ist, wenn n die Zahl der Elemente ist.

   (b) Zeigen Sie, dass zwei Fibonacci-Heaps mit n_1 und n_2 Elementen in $O(\log(n_1 + n_2))$ Zeit verschmolzen werden können. Das Ergebnis soll also ein Fibonacci-Heap sein, der alle $n_1 + n_2$ Elemente enthält. (2+2 Punkte)

4. Implementieren Sie Fibonacci-Heaps, um eine Menge von Objekten mit reellen Zahlen als Schlüsseln zu verwalten. Insbesondere soll Ihre Datenstruktur die Operationen Insert, Deletemin und Decreasekey unterstützen und dabei die in der Vorlesung genannten Laufzeiten erreichen. Benutzen Sie Ihre Datenstruktur für eine Implementierung von Prims Algorithmus, die in einer Laufzeit von $O(m + n \log n)$ einen kostenminimalen aufspannenden Baum in einem gegebenen ungerichteten zusammenhängenden Graphen mit n Knoten und m Kanten berechnet. Nähere Angaben zur Programmieraufgabe finden Sie auf der Rückseite. (15 Punkte)

Abgabe: Dienstag, den 7.11.2017, vor der Vorlesung.

Zur Programmieraufgabe:
Das Programm muss in C oder C++ geschrieben sein. Es wird empfohlen, C++ zu verwenden. In diesem Fall kann man zum Einlesen und Speichern der Graphen die Klasse `Graph` aus der Vorlesung "Algorithmische Mathematik I" aus dem Wintersemester 2016/2017 verwenden.
Alle Datenstrukturen, die in dieser Vorlesung vorgestellt wurden, finden Sie hier:

`http://www.or.uni-bonn.de/~hougardy/alma/alma.html`

Sie können alle diese Programme und Datenstrukturen verwenden. Insbesondere können Sie die dortige Implementierung von Prims Algorithmus (mit geeigneten Änderungen) benutzen. Außerdem dürfen Sie die STL verwenden, aber keine weiteren externen Bibliotheken.

Einlesen der Daten: Eine gültige Datei, die einen Graphen beschreibt, hat das folgende Format:

```
Knotenanzahl
Knoten0a Knoten0b Gewicht0
Knoten1a Knoten1b Gewicht1
...
```

In der ersten Zeilen steht eine einzelne natürliche Zahl, welche die Anzahl der Knoten angibt. Jede weitere Zeile spezifiziert genau eine Kante. Die ersten beiden Einträge einer Zeile sind zwei verschiedene nichtnegative ganze Zahlen, welche die Nummern der Endknoten der Kante sind. Dabei nehmen wir an, dass, wenn wir n Knoten haben, die Knoten von 0 bis $n-1$ durchnummeriert sind. Die Reihenfolge der beiden ersten Zeileneinträge spielt keine Rolle. Der dritte Eintrag in der Zeile ist eine reelle Zahl, die das Gewicht der Kante bezeichnet. Sie können annehmen, dass der Graph einfach ist und dass alle Kantengewichte in Variablen vom Typ `double` gespeichert werden können.
Die Einleseroutine, die Sie auf der oben genannten Homepage finden können, ist mit diesem Format kompatibel.

Ausgabeformat: In der Ausgabe des Programms sollen in jeder Zeile die zwei Endknoten einer Kante des berechneten Baumes geschrieben werden. Dabei soll keine Kante doppelt genannt werden.

Beispiel: Eine Eingabedatei für einen Graphen mit fünf Knoten und sechs Kanten kann so aussehen:

```
5
0 1 2.5
2 1 2
2 3 3.8
3 1 -2
0 2 -12.6
3 4 0
```

Die Ausgabe muss dann (bis auf Permutation der Zeilen und Vertauschung der Einträge innerhalb der Zeilen) so aussehen:

```
0 2
1 3
3 4
1 2
```

Das Programm muss korrekt arbeiten und ohne Fehlermeldung kompiliert werden können. Der Code muss auf einem gängigen Linuxsystem funktionieren.

Abgabe: Der Quelltext des Programms muss bis 7. November, 16:15 Uhr per E-Mail beim jeweiligen Tutor eingegangen sein. Außerdem ist bis zu diesem Zeitpunkt ein Ausdruck des Quelltextes zusammen mit den Theorieaufgaben abzugeben.